

# Spatial Point Patterns

Lecture #1

## Point pattern terminology

- ▶ **Point** is the term used for an arbitrary location
- ▶ **Event** is the term used for an observation
- ▶ **Mapped point pattern**: all relevant events in a study area  $R$  have been recorded
- ▶ **Sampled point pattern**: events are recorded from a sample of different areas within a region



## Objective of point pattern analysis

- ▶ Determine if there is a tendency of events to exhibit a systematic pattern over an area as opposed to being randomly distributed
- ▶ Point data often have attributes, but right now we are only interested in the location in point pattern analysis
- ▶ Does a pattern exhibit clustering or regularity?
- ▶ Over what spatial scales do patterns exist?

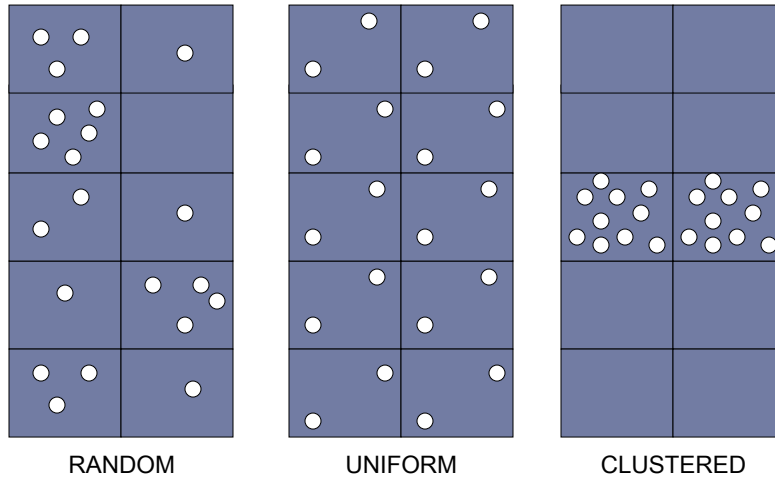


## Types of distributions

- ▶ Three general patterns
  - ▶ **Random** - any point is equally likely to occur at any location and the position of any point is not affected by the position of any other point
  - ▶ **Uniform** - every point is as far from all of its neighbors as possible
  - ▶ **Clustered** - many points are concentrated close together, and large areas that contain very few, if any, points



## Types of distributions



## Methods

- ▶ “Exploratory” analysis
  - ▶ Visualization (maps)
  - ▶ Estimate how intensity of point pattern varies over an area
    - ▶ Quadrat analysis, kernel estimation
  - ▶ Estimate the presence of spatial dependence among events
    - ▶ Nearest neighbor distances, K-function
- ▶ Modeling techniques
  - ▶ Statistical tests for significant spatial patterns in data, compared with the null hypothesis of complete spatial randomness (CSR)
- ▶ Much of the time we do both!



## How Bailey & Gatrell see it

---

- ▶ Exploring 1<sup>st</sup> order properties
  - ▶ Measuring intensity – based on the density (or mean number of events) in an area
    - ▶ Quadrat analysis
    - ▶ Kernel estimation
- ▶ Exploring 2<sup>nd</sup> order properties
  - ▶ Measuring spatial dependence – based on distances of points from one another
    - ▶ Nearest neighbor distances
    - ▶ K-function



## Modeling techniques

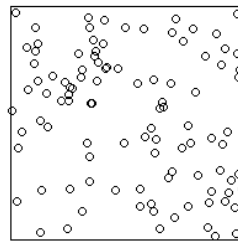
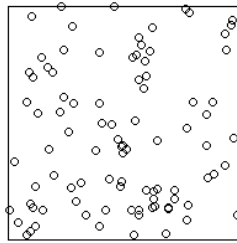
---

- ▶ We can conduct statistical tests for significant patterns in our data
  - ▶  $H_0$ : events exhibit complete spatial randomness (CSR)
  - ▶  $H_a$ : events are spatially clustered or dispersed
- ▶ What is complete spatial randomness?
- ▶ What are we comparing our point pattern to?



## Complete spatial randomness

- ▶ CSR assumes that points follow a *homogeneous Poisson process* over the study area
  - ▶ The density of points is constant (homogeneous) over the study area
  - ▶ For a random sample of subregions, the frequency distribution of the number of points in each region will follow a Poisson distribution
    - ▶ # of points in an given subregion is the same for all subregions in study area
    - ▶ # of points in a subregion independent of # of points in any other subregion



## Some notes on R

```
> library(maptools)
> library(rgdal)
> library(shapefiles)
> library(spatstat)
> library(splancs)

> workingDir = "C:/Users/Eroot/Quant/R"
```



## Splancs and Spatstat in R

- ▶ Use different data file formats for analysis
  - ▶ Both need a set of “points” and a study area “boundary”

### ▶ Splancs

```
> library(shapefiles)
> border <- readShapePoly(paste(workingDir,
  "/shapefiles/FLBndy.shp", sep=""))
> flbord <- border@polygons[[1]]@Polygons[[1]]@coords
> str(border)

> flinv<-readShapePoints("C:/Users/Elisabeth
  Root/Desktop/Quant/R/shapefiles/FL_Invasive.shp")
> flinvxy<-coordinates(flinv)
```



## Splancs and Spatstat in R

### ▶ Spatstat

```
> library(shapefiles)
> library(maptools)

> flinv<-
  readShapePoints("C:/Users/Eroot/Quant/R/shapefiles/
  FL_Invasive.shp")
> flpt<-as(flinv,"ppp")

> border <- readShapePoly(paste(workingDir,
  "/shapefiles/FLBndy.shp", sep=""))
> flbdry<-as(border,"owin")

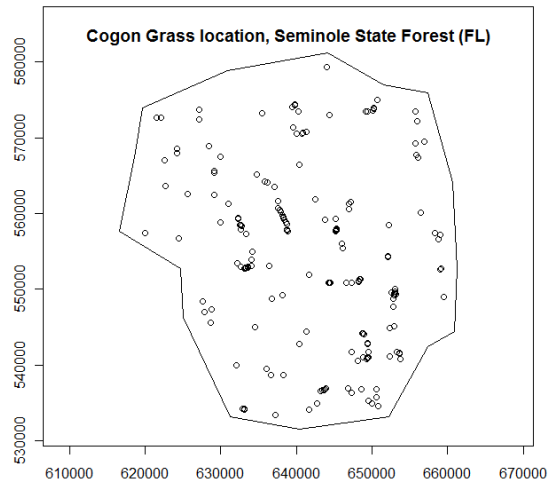
> flppp<-ppp(flpt$x, flpt$y, window=flbdry)
```



## Sample dataset plot

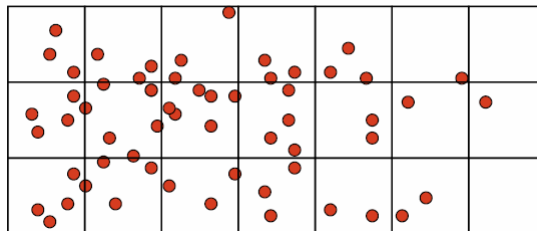
- ▶ Dataset: Location of Cogon Grass (invasive species in FL)

```
> plot(flppp, axes=T)
```



## Quadrat methods

- ▶ Divide the study area into subregions of equal size
  - ▶ Often squares, but don't have to be
- ▶ Count the frequency of events in each subregion
- ▶ Calculate the intensity of events in each subregion



## Quadrat methods

number	3	3	3	3	3	1	0
	4	6	6	4	2	1	1
	4	4	3	3	2	2	0

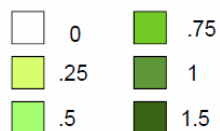
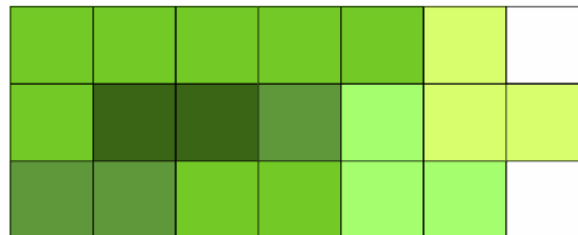
$\lambda = n/A$  where  $n$  = number of events and  $A = 4$  is area of each quadrat

Intensity $\lambda$	.75	.75	.75	.75	.75	.25	0
	1	1.5	1.5	1	.5	.25	.25
	1	1	.75	.75	.5	.5	0



## Quadrat method

- ▶ Compare the intensity variation over R



3	1
5	0
2	1
1	3
3	1

Quadrat	# of Points	x^2
#	Per Quadrat	
1	3	9
2	1	1
3	5	25
4	0	0
5	2	4
6	1	1
7	1	1
8	3	9
9	3	9
10	1	1
	20	60
Variance	2.222	
Mean	2.000	
Var/Mean	1.111	

2	2
2	2
2	2
2	2
2	2

Quadrat	# of Points	x^2
#	Per Quadrat	
1	2	4
2	2	4
3	2	4
4	2	4
5	2	4
6	2	4
7	2	4
8	2	4
9	2	4
10	2	4
	20	40
Variance	0.000	
Mean	2.000	
Var/Mean	0.000	

0	0
0	0
10	10
0	0
0	0

Quadrat	# of Points	x^2
#	Per Quadrat	
1	0	0
2	0	0
3	0	0
4	0	0
5	10	100
6	10	100
7	0	0
8	0	0
9	0	0
10	0	0
	20	200
Variance	17.778	
Mean	2.000	
Var/Mean	8.889	

N = number of quadrats = 10

$$\text{Variance} = \frac{\sum x^2 - [(\sum x)^2 / N]}{N - 1}$$

$$\text{Variance mean ratio} = \frac{\text{variance}}{\text{mean}}$$

To test for CSR, calculate the test statistic for quadrat ( $\chi^2$ ):

$$= \frac{(m-1)s^2}{\bar{x}}$$

m = # of quadrats  
s<sup>2</sup> = observed variance  
x = observed mean

Compare to  $\chi^2$  distribution with m-1 degrees of freedom

## Quadrats in R

### Done using spatstat package

```
> qt <- quadrat.test(flppp,
  nx = 10, ny = 10)
```

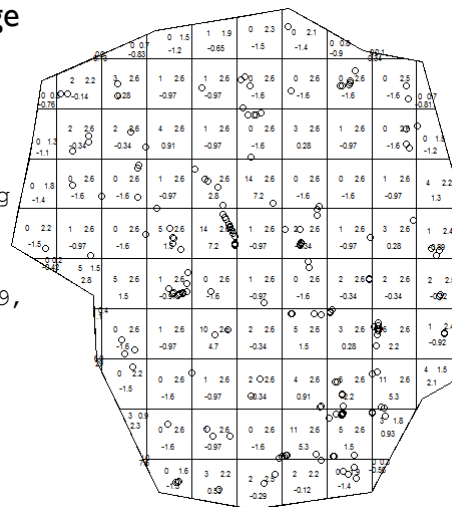
```
> qt
```

Chi-squared test of CSR using quadrat counts

```
X-squared = 1239.057, df = 89,
p-value < 2.2e-16
```

```
> plot(flppp)
```

```
> plot(qt, add = TRUE, cex =
  .5)
```



## Weaknesses of quadrat method

- ▶ **Quadrat size**
  - ▶ If too small, they may contain only a couple of points
  - ▶ If too large, they may contain too many points
- ▶ **Actually a measure of dispersion, and not really pattern,** because it is based primarily on the density of points, and not their arrangement in relation to one another
- ▶ **Results in a single measure for the entire distribution, so** variations within the region are not recognized

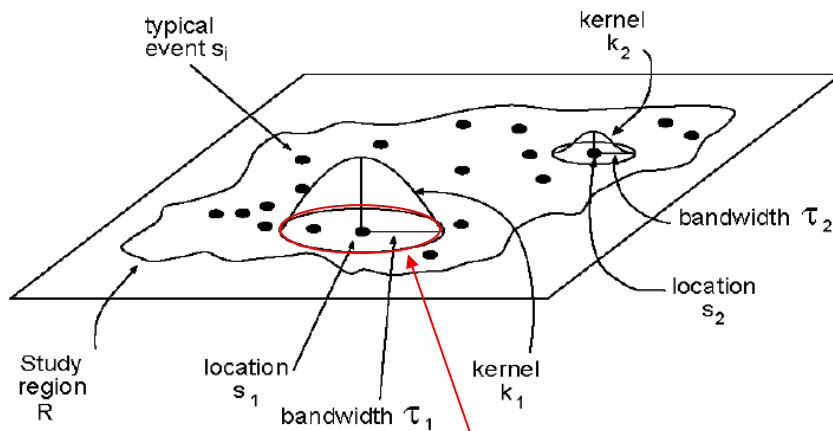


## Kernel estimation

- ▶ **Believe it or not, we already talked about this with GWR!**
- ▶ **Calculating the density of events within a specified search radius around each event**
  - ▶ A moving three-dimensional function (the kernel) of a given radius (bandwidth) 'visits' each point in the study area
  - ▶ Use kernel to weight the area surrounding the point proportionately to its distance to the event
  - ▶ Sum these individual kernels for the study region
  - ▶ Produce a smoothed surface
- ▶ **Variety of different kernels**
  - ▶ Bivariate quartic most common



## Kernel estimation



- Creating a smooth surface for each kernel
- Surface value highest in the center (point location) and diminishes with distance...reaches 0 at radius distance

## Kernel estimation

- ▶  $s$  is a location in  $R$  (the study area)
- ▶  $s_1 \dots s_n$  are the locations of  $n$  events in  $R$
- ▶ The intensity at a specific location is estimated by:

$$\hat{\lambda}_\tau(s) = \sum_{i=1}^n \frac{1}{\tau^2} k\left(\frac{s - s_i}{\tau}\right)$$

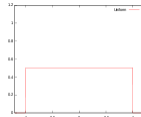
Annotations for the equation:

- distance between point  $s$  and  $s_i$  (points to  $s - s_i$ )
- bandwidth (radius of the circle) (points to  $\tau$ )
- kernel (which is a function of the distance and bandwidth) (points to the  $k(\cdot)$  function)

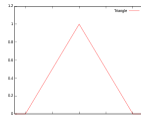
- ▶ Summed across all points  $s_i$  within the radius ( $\tau$ )

## Different types of kernels

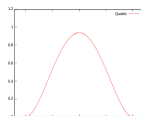
Uniform



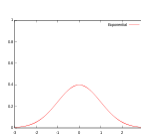
Triangular



Quartic



Gaussian



$$\hat{\lambda}_\tau(s) = \sum_{i=1}^n \frac{1}{\tau^2} k\left(\frac{s - s_i}{\tau}\right)$$

Each kernel type has a different equation for the function  $k$ , for example:

Triangular:  $k = 1 - \left| \frac{d_i}{\tau} \right|$

Quartic:  $k = \frac{3}{\pi} \left( 1 - \frac{h_i^2}{\tau^2} \right)$

Normal:  $k = \frac{1}{\sqrt{2\pi}} e^{-\frac{h_i^2}{2\tau^2}}$

## Kernel estimation

- ▶ The kernel ( $k$ ) is basically a mathematical function that calculates how the surface value “falls off” as it reaches the radius
- ▶ There are lots of different kernel functions
  - ▶ Most researchers believe it doesn't really matter which you use
  - ▶ Most common in GIS is the quartic kernel

$$\hat{\lambda}_\tau(s) = \sum_{d_i < \tau} \frac{3}{\pi \tau^2} \left( 1 - \frac{d_i^2}{\tau^2} \right)^2$$

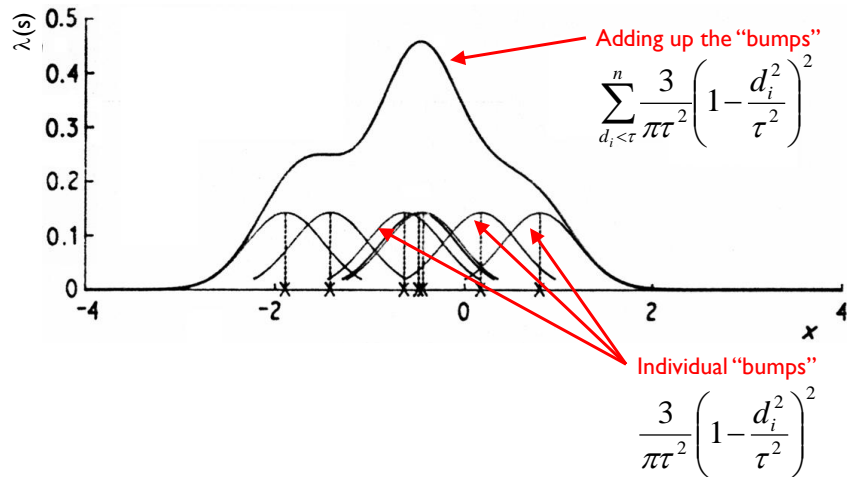
distance between point  $s$  and  $s_i$

bandwidth (radius of the circle)

At point  $s$ , the weight is  $3/\pi\tau^2$  and drops smoothly to a value of 0 at  $\tau$

- ▶ Summed for all values of  $d_i$  which are not larger than  $\tau$

## Kernel estimation



## A few notes

- ▶ Like GWR, we can use fixed and adaptive kernels
  - ▶ Fixed = bandwidth is a specified distance
  - ▶ Adaptive = fixed number of points used
- ▶ Results are sensitive to change in bandwidth
  - ▶ When bandwidth is larger, the intensity will appear smooth and local details obscured
  - ▶ When bandwidth is small, the intensity appears as local spikes at event locations
  - ▶ No agreement on how to select the "best" bandwidth
    - ▶ prior information about underlying spatial process
    - ▶ comparison of various bandwidths
    - ▶ using Mean Square Error (in R)

## Kernel estimation in R

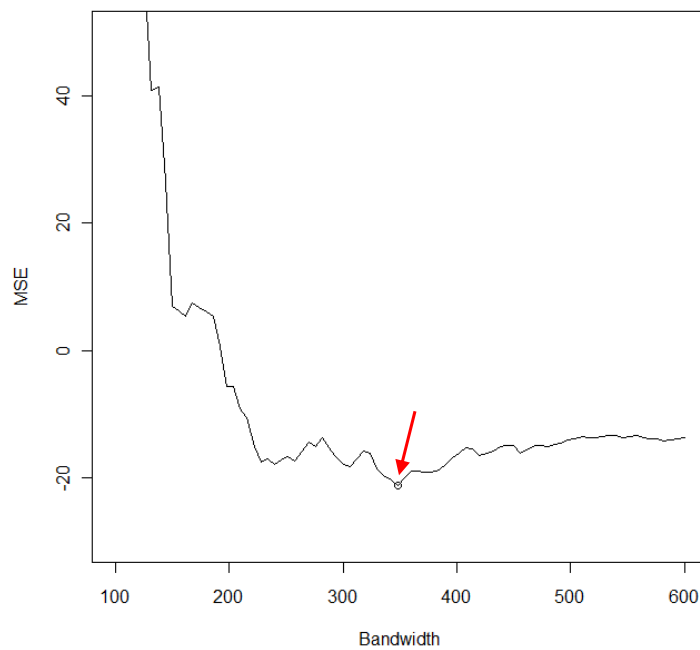
- ▶ Can be done in both `splancs` and `spatstat`
  - ▶ `splancs` = quartic kernel
  - ▶ `spatstat`=gaussian kernel
- ▶ Mean standard error one way to find “optimal bandwidth”

```
> mse<-mse2d(flinvxy,flbord, 100, 600)

> plot(mse$h, mse$mse, xlab="Bandwidth", ylab="MSE",
      type="l", xlim=c(100,600), ylim=c(-30,50))

> i<-which.min(mse$mse)

> points(mse$h[i], mse$mse[i])
```



## Kernel estimation in R

- ▶ Need to make a grid to “dump” kernel estimates into
  - ▶ The `Sobj_SpatialGrid()` function in `maptools` takes a `maxDim=` argument, which indirectly controls the cell resolution

```
> sG <- Sobj_SpatialGrid(border, maxDim=400)$SG
> grd <- slot(sG, "grid")
> summary(grd)
```

- ▶ Can also create a `GridTopology` object from scratch:

```
> poly <- slot(border, "polygons")[[1]]
> poly1 <- slot(poly, "Polygons")[[1]]
> coords <- slot(poly1, "coords")
> min(coords[,1])
> min(coords[,2])
> grd <- GridTopology(cellcentre.offset=c(616593,531501),
  cellsize=c(150,150), cells.dim=c(400,400))
> summary(grd)
```



## Kernel estimation in R

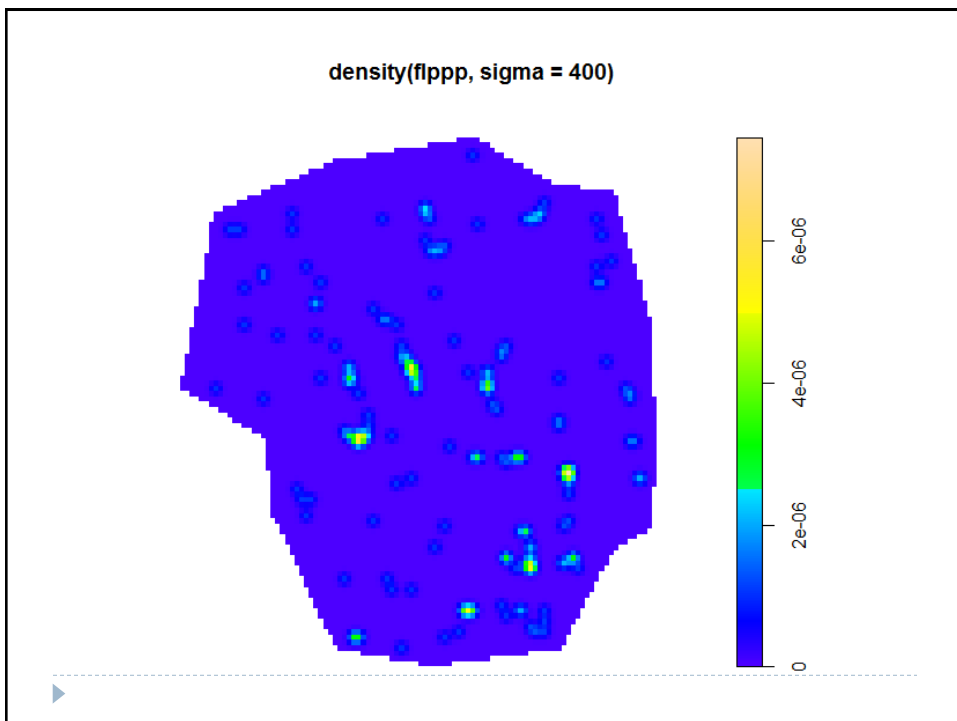
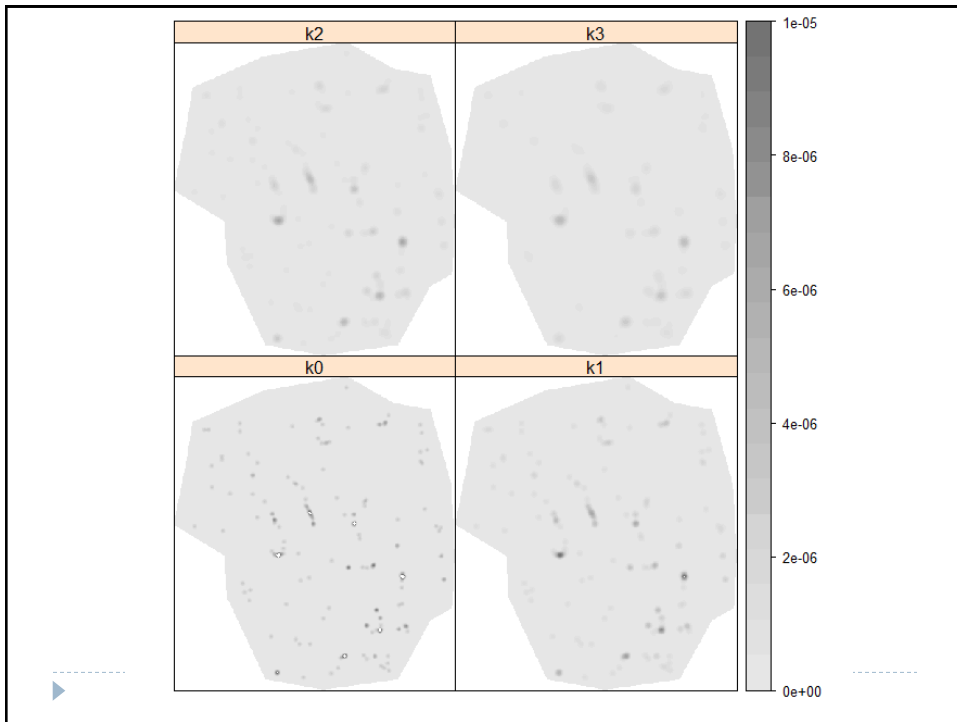
- ▶ Using `splancs`

```
> k0 <- spkernel2d(flinvxy, flbord, h0=400, grd)
> k1 <- spkernel2d(flinvxy, flbord, h0=600, grd)
> k2 <- spkernel2d(flinvxy, flbord, h0=800, grd)
> k3 <- spkernel2d(flinvxy, flbord, h0=1000, grd)
> df <- data.frame(k0=k0, k1=k1, k2=k2, k3=k3)
> kernels <- SpatialGridDataFrame(grd, data=df)
> summary(kernels)
> gp <- grey.colors(5, 0.9, 0.45, 2.2)
> print(spplot(kernels, at=seq(0,.00001,length.out=20),
  col.regions=colorRampPalette(gp)(21)))
```

- ▶ Using `spatstat`

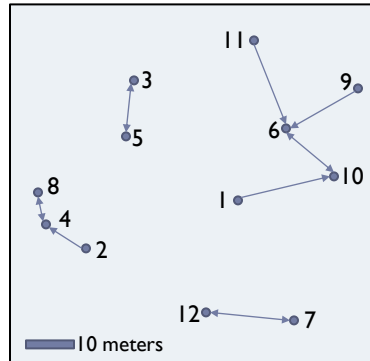
```
> plot(density(flppp, sigma = 600))
```





## Nearest neighbor analysis G-function

- ▶ Simplest measure and is similar to the mean
- ▶ Examine the cumulative frequency distribution of the nearest neighbor distances

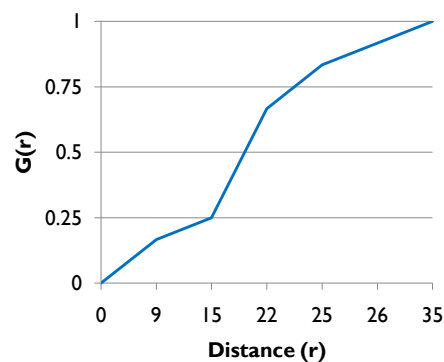


Event	x	y	Nearest neighbor	$r_{\min}$
1	66.22	32.54	10	25.59
2	22.52	22.39	4	15.64
3	31.01	81.21	5	21.14
4	9.47	31.02	8	24.81
5	30.78	60.10	3	9.00
6	75.21	58.93	10	21.14
7	79.26	7.68	12	21.94
8	8.23	39.93	4	9.00
9	98.73	42.53	6	21.94
10	89.78	42.53	6	21.94
11	65.19	92.08	6	34.63
12	54.46	8.48	7	24.81



## G-function

Event	x	y	Nearest neighbor	$r_{\min}$
1	66.22	32.54	10	25.59
2	22.52	22.39	4	15.64
3	31.01	81.21	5	21.14
4	9.47	31.02	8	24.81
5	30.78	60.10	3	9.00
6	75.21	58.93	10	21.14
7	79.26	7.68	12	21.94
8	8.23	39.93	4	9.00
9	98.73	42.53	6	21.94
10	89.78	42.53	6	21.94
11	65.19	92.08	6	34.63
12	54.46	8.48	7	24.81



$$G(r) = \frac{\# [r_{\min}(s_i) < r]}{n}$$

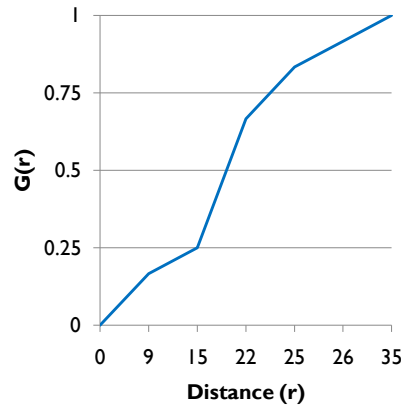
$$= \frac{\# \text{ point pairs where } r_{\min} \leq r}{\# \text{ of points in study area}}$$



## G-function

The shape of G-function tells us the way the events are spaced in a point pattern

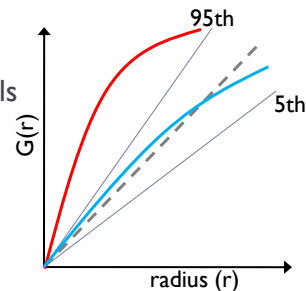
- ▶ Clustered = G increases rapidly at short distance
- ▶ Evenness = G increases slowly up to distance where most events spaced, then increases rapidly
- ▶ How do we examine significance (significant departure from CSR)?



▶

## How do we tell if G is significant?

- ▶ The significance of any departures from CSR (either clustering or regularity) can be evaluated using simulated “confidence envelopes”
- ▶ Simulate many (1000??) spatial point processes and estimate the G function for each of these
  - ▶ Rank all the simulations
  - ▶ Pull out the 5<sup>th</sup> and 95<sup>th</sup> G(r) values
  - ▶ Plot these as the 95% confidence intervals
  - ▶ This is done in R!



▶

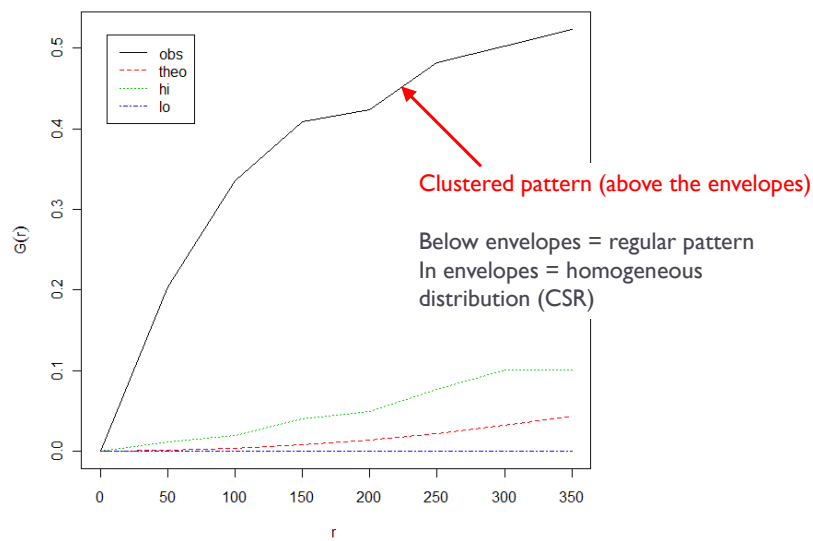
## G estimate in R

```
> r=seq(0,350,by=50)

> G <- envelope(flppp, Gest, r=r, nsim = 59, rank = 2)
> G
Pointwise critical envelopes for G(r)
Edge correction: "km"
Obtained from 59 simulations of CSR
Significance level of pointwise Monte Carlo test: 2/60 = 0.03333
Data: flppp
Entries:
id      label      description
--      -
r       r          distance argument r
obs     obs(r)      observed value of G(r) for data pattern
theo    theo(r)     theoretical value of G(r) for CSR
lo      lo(r)       lower pointwise envelope of G(r) from simulations
hi      hi(r)       upper pointwise envelope of G(r) from simulations
> plot(G)
```



## G estimate in R



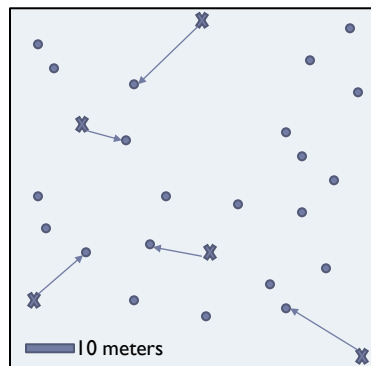
## Nearest neighbor analysis

### F-function

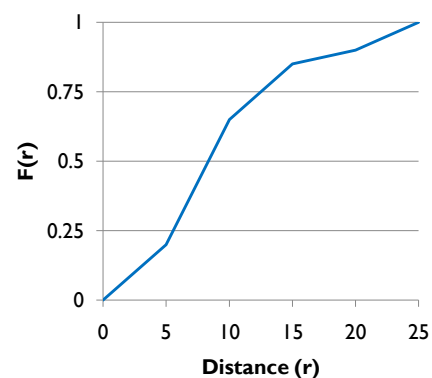
- ▶ Select a sample of point locations anywhere in the study region at random
  - ▶ Determine minimum distance from each point to any event in the study area
- ▶ Three steps:
  1. Randomly select  $m$  points ( $p_1, p_2, \dots, p_m$ )
  2. Calculate  $d_{\min}(p_i, s)$  as the minimum distance from location  $p_i$  to any event in the point pattern  $s$
  3. Calculate  $F(d)$



### F-function



- ✕ = randomly chosen point
- = event in study area
- =  $d_{\min}$



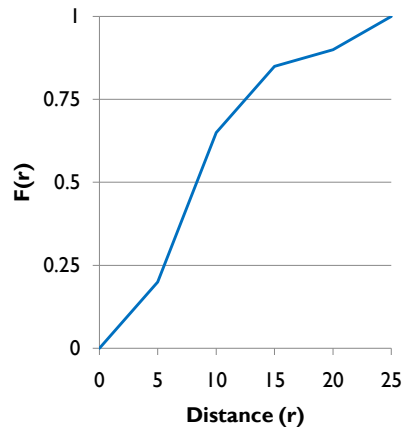
$$F(d) = \frac{\#[d_{\min}(p_i, s) < d]}{m}$$

$$= \frac{\text{\# of point pairs where } r_{\min} \leq r}{\text{\# sample points}}$$



## F-function

- ▶ Clustered =  $F(r)$  rises slowly at first, but more rapidly at longer distances
- ▶ Evenness =  $F(r)$  rises rapidly at first, then slowly at longer distances
- ▶ Examine significance by simulating “envelopes”



## F estimate in R

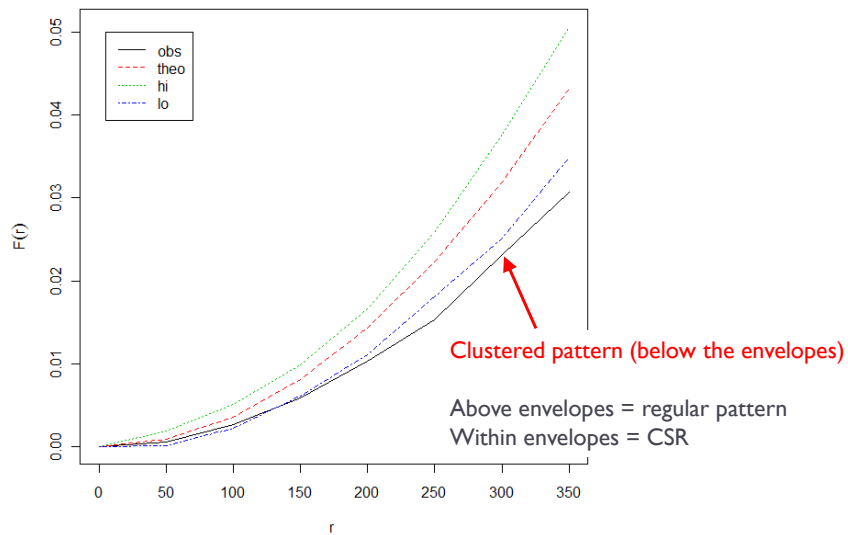
```
> r=seq(0,350,by=50)
```

```
> F <- envelope(flppp, Fest, r=r, nsim = 59, rank = 2)
```

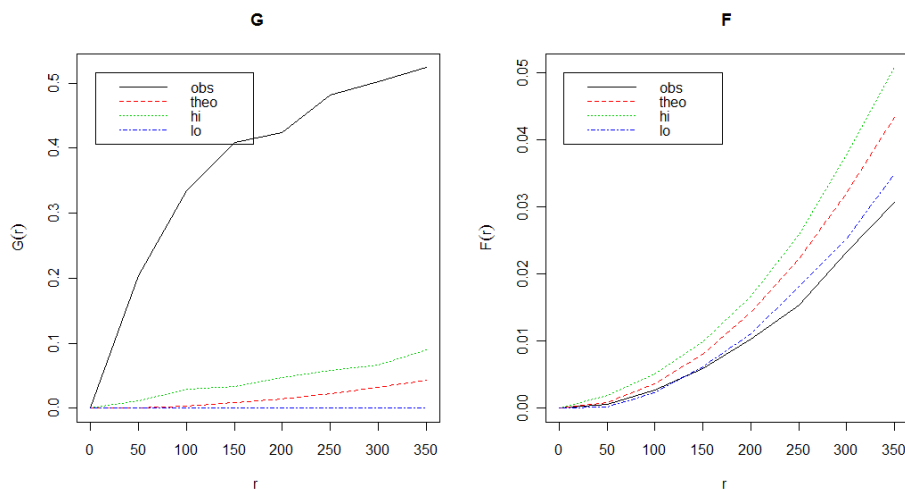
```
> plot(F)
```

	lty	col	key	label	meaning
obs	1	1	obs	obs(r)	observed value of $F(r)$ for data pattern
theo	2	2	theo	theo(r)	theoretical value of $F(r)$ for CSR
hi	3	3	hi	hi(r)	upper pointwise envelope of $F(r)$ from simulations
lo	4	4	lo	lo(r)	lower pointwise envelope of $F(r)$ from simulations

## F estimate in R



## Comparison between G and F



## K function

---

- ▶ Limitation of nearest neighbor distance method is that it uses only nearest distance
    - ▶ Considers only the shortest scales of variation
  - ▶ K function (Ripley, 1976) uses more points
    - ▶ Provides an estimate of spatial dependence over a wider range of scales
    - ▶ Based on all the distances between events in the study area
    - ▶ Assumes isotropy over the region
- 



## K function

---

- ▶ Defined as:
 
$$K(h) = \frac{1}{\lambda} E(\#(\text{events w/in distance } h \text{ of randomly chosen event}))$$
  - ▶  $\lambda$  = the intensity of events ( $n/A$ )
- 



## How do we estimate the K-function

1. Construct a circle of radius  $h$  around each point event (i)
2. Count the number of other events (j) that fall inside this circle
3. Repeat these two steps for all points (i) and sum results

$$\hat{K}(h) = \frac{R}{n^2} \sum_{i \neq j} \frac{I_h(d_{ij})}{w_{ij}}$$

area of  $R$  →  
 number of points →  $n^2$   
 dummy variable  
 1 if  $d_{ij} \leq h$   
 0 otherwise  
 edge correction  
 the proportion of circumference of circle  
 (centered on point i, containing point j)  
 = 1 if whole circle in the study area

4. Increment  $h$  by a small amount and repeat the computation

## Interpreting the K-function

- ▶  $K(h)$  can be plotted against different values of  $h$
- ▶ But what should  $K$  look like for no spatial dependence?
- ▶ Consider what  $K(h)$  should look like for a random point process (CSR)
  - ▶ The probability of an event at any point in  $R$  is independent of what other events have occurred **and** equally likely anywhere in  $R$

## Interpreting the K function

- Under the assumption of CSR, the expected number of events within distance  $h$  of an event is:

$$K(h) = \pi h^2$$

← the radius of the circle
 The density of events should be evenly distributed across all circles

- $K(h) < \pi h^2$  if point pattern is regular
  - $K(h) > \pi h^2$  if point pattern is clustered
- Now we can compare  $K(h)$  to  $\pi h^2$ 
  - How do we do this?

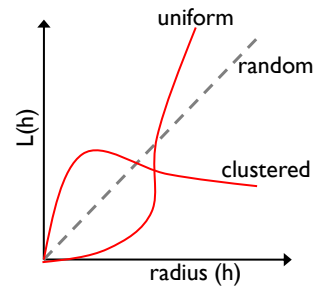


## Interpreting K with L

- This L-function is nothing more than a standardized version of the K function
  - Transforms the K function so we can easily interpret it
    - Compare it to 0

$$\hat{L}(h) = \sqrt{\frac{\hat{K}(h)}{\pi}} - h$$

- $L(h) = 0$  if point process is random
  - Peaks of positive values = clustering
  - Troughs of negative values = regularity
- Significance of any departures from  $L=0$  evaluated using simulated “confidence envelopes”

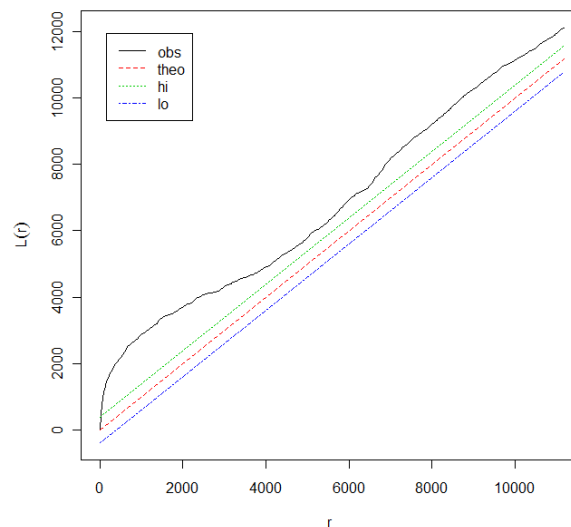


## K function in R

```
> L <- envelope(flppp, Lest, nsim = 59, rank = 2, global=TRUE)
> L
Simultaneous critical envelopes for L(r)
Edge correction: "iso"
Obtained from 59 simulations of CSR
Significance level of Monte Carlo test: 1/60 = 0.0166667
Data: flppp
Entries:
id      label      description
--      -
r        r        distance argument r
obs      obs(r)     observed value of L(r) for data pattern
theo     theo(r)     theoretical value of L(r) for CSR
lo       lo(r)     lower critical boundary for L(r)
hi       hi(r)     upper critical boundary for L(r)
> plot(L)
```



## K function in R



## Real world situations

---

- ▶ In the real world, the location of events is often related to underlying patterns
  - ▶ Population centers
  - ▶ Events that may not seem to cluster in space, but cluster in space time
- ▶ There are many (many many) variations of point pattern analysis
- ▶ Often called “multivariate point pattern” analysis
  - ▶ Comparing distributions of multiple sets of points

---

▶